

# Agenda - Beispiel

Dies ist unsere empfohlene Agenda für das Thema Angular for Enterprise. Wir verfügen darüber hinaus über weiteres Kursmaterial, um Themen einfach auszutauschen.

## Architektur großer Anwendungen

Wie teilt man eine Applikation am Besten auf und warum? Wie programmiere ich zukunftssicher? In diesem Teil liefern wir die Antworten.

- Schneiden von Modulen
- Wiederverwendbare Pakete
- Arbeiten mit Monorepos
- Angular Elements
- Web Components

## RxJS für Reaktive Architekturen

Was ist der Unterschied zwischen ReplaySubject und BehaviorSubject und wann benutze ich was? Wie benutze ich RxJS(Reaktive Erweiterungen für JavaScript) um meine Anwendung wartbar zu implementieren?

- Observables
- Cold vs. Hot Observables
- Operatoren im Detail
- Observables vs. Subjects
- Different Types of Subjects

## Testing & Debugging

Mit dem Wachsen einer Applikation wird das manuelle Testen immer aufwändiger bis fast unmöglich. Wir zeigen - wann man welche Tests einsetzt, wie man testet und Best Practices.

- Unit Tests vs. End-to-End Tests
- Unit Testing
- End-To-End Testing mit Protractor
- Source Maps
- Augury
- DevTools

## **Routing für Fortgeschrittene**

Wir beschäftigen uns hier intensiv mit dem Lazy-loading von Modulen. Vor allen Dingen größere Applikationen profitieren enorm davon, wenn man am Anfang erst einen kleinen Kern an den Benutzer ausliefert und weitere Teile der Applikation später nachlädt.

- Aufteilung in Module
- Lazy-Loading
- Möglichkeiten des Router-Event-Systems
- Guards
- Resolver
- Lazy-Loading und Preloading

## **Statenmanagement in Angular**

Bei großen Anwendungen mit vielen Komponenten ist es oft schwer, State-Änderungen zu verfolgen. Bei tiefen Verschachtelungen werden meistens zu viele Daten an Komponenten weitergegeben - was die Struktur komplexer macht. Redux ist eine Alternative dazu.

- State-Libraries im Vergleich
- Redux und @ngrx/Store
- @ngrx/Effects für asynchrone Operationen nutzen
- Performanceverbesserung mit Immutables
- Observables mit Redux nutzen

## **Performance**

Der Titel spricht für sich. Wir verkürzen wir die Ladezeiten durch kleinere Pakete? Wie können wir Daten auf dem Client zwischenspeichern? Wie können wir aufwändigen Programmcode beschleunigen?

- AOT-Kompilierung und Tree-Shaking
- Lazy Loading und Preloading
- Caching mit Service Worker
- Rendering im Hintergrund mit Web Worker